(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 January 2002 (31.01.2002)

PCT

(10) International Publication Number
WO 02/09350 A2

(51) International Patent Classification[7]: H04L 12/00

(21) International Application Number: PCT/EP01/08552

(22) International Filing Date: 20 July 2001 (20.07.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/616,632 26 July 2000 (26.07.2000) US

(71) Applicant: **KONINKLIJKE PHILIPS ELECTRONICS N.V.** [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(72) Inventors: **MOONEN, Jan, R.**; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL). **SHTEYN, Yevgeniy, E.**; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(74) Agent: **GROENENDAAL, Antonius, W., M.**; Internationaal Octrooibureau B.V., Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(81) Designated States *(national)*: CN, JP, KR.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/09350 A2

(54) Title: SERVER-BASED MULTI-STANDARD HOME NETWORK BRIDGING

(57) Abstract: A bridge in a home network couples first and second clusters of devices. The clusters have different software architectures. The bridge is connected to a server on the Internet. This server offers a lookup service for some set of standards, and allows a bridge to locate and download the appropriate translation modules for allowing a device in the first cluster to interact with the second cluster.

Server-based multi-standard home network bridging

This application is a Continuation-in-Part of U.S. serial no. 09/340,272 (attorney docket PHA 23,634) filed 6/25/99 for Yevgeniy Eugene Shteyn for BRIDGING MULTIPLE HOME NETWORK SOFTWARE ARCHITECTURES.

The invention relates to the bridging of multiple networks based on different software architectures. The invention relates in particular to home networking.

It seems to be unlikely that there will ever be a single universally applicable networking standard for each device in the home. Multiple standards for software architectures coexist and new ones will emerge. New standard interfaces will be developed for new types of devices that are specifically targeted by those standards. Home applications are designed to make an intelligent use of all devices available in the home, but are not capable of dealing with each and every networking standard in use or becoming available in the future. Similarly, devices themselves will not be able to support every existing home networking standard. For these reasons, bridges are needed between the different sub-networks or clusters, each respective one complying with a specific respective standard. A bridge serves to transparently represent a device, complying with a first standard in a network cluster of a first type, as a device complying with a second standard in a network cluster of a second type. The result is a single unified view of the home network available to software applications written for the first standard and as well those written for the second standard.

Typically, a device on a network is controlled through a set of messages complying with the interface of the device. Interoperability between devices and software applications depends upon standard interfaces with a unique identification. Once an application knows the unique identification of the device, it knows the interface of the device and it can control the device by sending it messages. To the application it makes no difference whether it is sending standard messages directly to the device itself, or indirectly via a software component that translates these messages into a different set of message that (eventually) achieve the same desired effect or state change at the controlled device. Hence, a bridge between networks of multiple different standards can be considered a multi-standard device. That is, the bridge complies with each of these multiple standards, and hosts software components that translate device interfaces from a first to a second standard and vice versa.

For example, a bridge is used between a cluster of five devices complying with a standard A and three devices complying with a standard B, different from A. The bridge hosts three translation modules for translating from A to B and five translation modules for translating from B to A. Accordingly, a software application capable of interacting with only standard A devices can now control all eight devices.

U.S. serial no. 09/340,272 (attorney docket PHA 23,634) filed 6/25/99 for Yevgeniy Eugene Shteyn for BRIDGING MULTIPLE HOME NETWORK SOFTWARE ARCHITECTURES, herein incorporated by reference, relates to the bridging of home networks of different software architectures. References to software representations of devices and services on a first one of the networks are automatically created. The references are semantically sufficient to enable automatic creation of at least partly functionally equivalent software representations for a second one of the networks so as to make the devices and services of the first network accessible from the second network. This document also addresses the HAVi, Home API and Jini software architectures.

In the following, the expression "translation module" includes the concept of "software representation", i.e., the representation in software of a physical device or of a service on a network or a sub-set thereof so as to make the device or service accessible to the relevant messaging or controlling software.

Preferably, a bridge performs the following functions: detection of the addition of a device in either of the bridged networks; identification of the type of the added device; locating the translation module for the identified device type if the device is likely to be of interest to the other network; and installing the translation module on the other network according to the procedure required by the standard used by that network.

Successful standards will continue to define standard interfaces for newly developed devices in a domain considered relevant to those standards. This necessitates the development of accompanying translation modules that enable those new devices to be represented in networks based on different standards. As a consequence, a bridge is unable to contain all embedded translation modules for all possible, relevant devices that become available in the future.

Accordingly, the inventors propose a solution wherein a bridge is connected to a server, e.g., on the Internet. This server offers a lookup service for some set of standards, and allows a bridge to locate and download the appropriate translation modules for use in the home network.

More specifically, the invention relates to a method of providing a service to a user of a home network. The method comprises enabling a component of a first cluster in the network to interact with a second cluster of the home network. The first cluster has a first software architecture, and the second cluster has a second software architectures different from the first one. The first and second clusters are coupled through a bridge. The method comprises enabling a server external to the clusters, e.g., on the Internet, to receive a reference of a component having a first software representation in the first cluster. The bridge may provide this reference. The method further comprises enabling to provide to the bridge a translation module, associated with the reference, for at least partially representing the component on the second cluster upon the module being installed on the bridge.

A service provider thus can maintain and update a data base of translation modules for any multiple standards being used in home networks. This partitioning and delegation of functionalities has many advantages as discussed below.

The invention allows the bridge to be fairly "light-weight" or low-cost, as it does not need to have embedded translation modules for all possible devices of all standards that it could be connected to in the home. Storage and computation power is only needed for the devices that are actually bridged (i.e., storage is not needed for the ones that are *potentially* bridged), and only *when* they are bridged (e.g., "just-in-time"-bridging, i.e., at the time of connecting the new device to the network).

Further, the invention renders the home network as a whole extensible and future-proof. As new devices are invented and descriptions thereof become part of various standard specs, these descriptions are translated and uploaded to the bridge server by, e.g., the device manufacturer or a third party, to make them available for bridging in existing home networks. This process does not require any update mechanism of components in the home network itself.

A further benefit is that the bridge-server operator is able to obtain information about the configuration of the individual user's home network. This information can be used to the advance of both the user and the manufacturers and service providers. See, for example, U.S. Serial No. 09/160,490 (attorney docket PHA 23,500) filed 9/25/1998 for Adrian Turner et al., for CUSTOMIZED UPGRADING OF INTERNET-ENABLED DEVICES BASED ON USER-PROFILE, incorporated herein by reference. This document relates to a server system that maintains a user profile of a particular end-user of consumer electronics network-enabled equipment and a data base of new technical features for this type of equipment, e.g., a home network. If there is a match between the user-profile and a new

technical feature, and the user indicates to receive information about updates or sales offers, the user gets notified via the network of the option to obtain the feature. Also see, e.g., U.S Serial No.09/189,535 (attorney docket PHA 23,527) filed 11/10/98 for Yevgeniy Shteyn for UPGRADING OF SYNERGETIC ASPECTS OF HOME NETWORKS, incorporated herein by reference. This document relates to a system with a server that has access to an inventory of devices and capabilities on a user's home network. The inventory is, for example, a look-up service as provided by HAVi, Jini and Home API architectures. The server has also access to a data base with information of features for a network. The server determines if the synergy of the apparatus present on the user's network can be enhanced based on the listing of the inventory and on the user's profile. If there are features that are relevant to the synergy, based on these criteria, the user gets notified. In this sense, U.S. serial no. 09/189,535 relates to the concept of an "application suggestor".

A further advantage of the invention resides in the fact that the server operator is enabled to measure market demands for particular devices to be bridged to a specific standard. The device manufacturer or another relevant third party can be notified of the emerging demand. When the new translation modules become available on the server, bridges that have sent requests for translation modules in the past with which the server could not comply, can now be notified of an upgrade.

Note that the bridge can be implemented as a software component on a device of a specific cluster on the home network in order to provide a bridge to another cluster. For example, a HAVi set top box can have a software component for bridging the HAVi cluster to, e.g., a UPnP cluster on the network. Similarly, a PC that controls the UPnP cluster can have a software component that serves to bridge the UPnP cluster of the home network to the HAVi cluster.

The invention is explained in further detail, and by way of example, with reference to the accompanying drawing, wherein:

Fig.1 is a block diagram illustrating the principle of the bridging between two networks according to the invention;

Fig.2 is a block diagram illustrating bridging HAVi to UPnP; and

Fig.3 is a block diagram illustrating bridging UPnP to HAVi.

Throughout the drawing, same reference numerals indicate similar or corresponding features.

As mentioned above an aspect of the invention relates to connecting a bridge to a server, e.g., on the Internet. This server offers a lookup service for some set of standards, and allows a bridge to locate and download the appropriate translation modules in the home network that eventually enables a device on a sub-network of a first architecture to work with devices on a sub-network of a second architecture.

Fig.1 is a diagram of a home network system 100 with a first cluster 102 of devices 104, 106 and 108 that comply with a first software architecture standard, herein-after called standard A. System 100 comprises a second cluster 110 of devices 112, 114 and 116 that comply with a second software architecture standard, herein-after called standard B. Clusters 102 and 110 are interconnected through a bridge 118. In order to have a meaningful network interaction between cluster 102 of standard A on the one hand, and cluster 110 of standard B on the other hand, translation modules are introduced. These modules need to participate in both clusters 102 and 110. The modules typically need components local to the clusters, such as lower-level communication software, in order to be capable of this participating. Rather than having each translation module comprising its own communication software it is more efficient to provide this software as an element of platform components 120 of bridge 118.

The process of the invention is now illustrated with an example wherein B-device 116 is going to be added to system 100.

The first step comprises physically connecting B-device 116 to B-cluster 110, or "booting" B-device 116.

In a next step, bridge 118 detects B-device 116 as a new addition, either because bridge 118 scans B-cluster 110 or its registry/directory/look-up service (not shown) periodically or because B-cluster 110 actively notifies bridge 118. Bridge 118 comprises a software component 122, referred to as Installation Manager, that handles the installation of further software components needed to integrate B-device 116 into system 100. An aspect hereof is discussed in, e.g., U.S. serial no. 09/340,272 (attorney docket PHA 23,634). In the latter document a software component, referred to as the Reference Factory, is capable of extracting information from any of the software representations of devices registered. This Reference Factory is capable of querying the inventory of services or of getting notified of a new software representation according to the methods of the relevant software architecture. Similarly, Installation Manager 122 receives or retrieves information descriptive of newly

added B-device 116. The descriptive information is possibly reformatted before being sent to a bridge server 124 via the Internet 126. In addition, bridge 118 preferably provides information about the local execution environment of home network 100. This information is relevant to the software components that server 124 downloads onto bridge 118. The relevant information regarding the environment relates to the software architectures present, in this case the A-standard cluster 102 and the B-standard cluster 110. The information may also relate to the memory available, the type of operating system(s) being used, virtual machines present, platform libraries, etc., on bridge 118. Based on this information server 124 is able to select the proper translation module or modules that fits or fit in best with the network environment of system 100.

Upon receipt of the descriptive and environment information, server 124 uses a look-up service that needs to match the information, descriptive of B-device 116, with a translation module for representing B-device 116 in A-cluster 102. In general, server 124 has a plurality of look-up services available: one for each ordered pair (X,Y), wherein X and Y are the standards supported by server 124. In order to support bi-directional bridging between a cluster of standard X and another cluster of standard Y, two look-up services are needed: (X,Y) and (Y,X). For support of bi-directional bridging between three clusters, all with different standards P, Q and R, six look-up services are needed: (P,Q); (Q,P); (P,R); (R,P); (Q,R); and (R,Q). Of course, server 124 may only support uni-directional bridging.

Devices, such as devices 104-108 and 112-116, are often composite objects. For example, a TV set typically comprises Display, Amplifier and Tuner components. Server 124 could first try to translate the composite object as a whole into a new composite device with an equivalent functionality. If that does not succeed server 124 could translate as much subsidiary components as possible on a one-to-one basis. This then would result in a partial, but still useful, mapping. For example, if A-cluster 102 has not available a definition of a Tuner, a B-type TV could still be bridged to A-cluster 102 as a monitor-like display/amplifier device. If a one-to-one relationship does not exist between particular subsidiary components in standard A and standard B, then a one-to-many or many-to-many mapping could be used. For example, standard A might define volume control and equalizer effects as a single subsidiary component, whereas standard B distinguishes them as separate subsidiary components. In this case, a device in B-cluster 110 that contains just the volume control component, but not the equalizer component, cannot be bridged to A-cluster 102. On the other hand, a device in A-cluster 102 that contains a single amplifier component (volume control as well as equalizer) can be bridged to network B by applying a one-to-two mapping

of the subsidiary components. In the most general case, a particular set of subsidiary components in A-cluster 102 matches with another set of subsidiary components in B-cluster 110 under a many-to-many mapping.

Next, assume that a matching translation module 128 has been found it is downloaded to the bridge, installed on platform 120 and registered in accordance with the protocol of standard A. This enables other applications and devices of A-cluster 102 to discover and use device 116 through module 128. The installation and registering of module 128 may be postponed until after it has been run on the execution environment of bridge 118.

The invention is explained below with an example illustrating the bridging of HAVi and Universal Plug and Play (UPnP) home networks with reference to Figs.2 and 3. The HAVi, Home API, and Jini standards for software architectures in the home networking field have been discussed to some detail in U.S. serial no. 09/340,272 (attorney docket PHA 23,634) mentioned above and incorporated herein by reference. In HAVi, a DCM (Device Control Module) is a software element that represents a single device or functionality on the HAVi network. The DCM exposes the HAVi defined APIs for that device. DCMs are dynamic in nature: if a device is inserted or removed from the network, a DCM for that device needs to be installed or removed, respectively, in the network. DCMs are central to the HAVi concept and the source of flexibility in accommodating new devices and features into the HAVi network.

Universal Plug and Play (UPnP) is an open network architecture that is designed to enable simple, ad hoc communication among distributed devices and software applications from multiple vendors. UPnP leverages Internet technology and extends it for use in non-supervised home networks. UPnP aims at controlling home appliances, including home automation, audio/video, printers, smart phones, etc. UPnP distinguishes between Control Points (CPs) and controlled devices (CDs). CPs comprise, e.g., browsers running on PCs, wireless pads, etc., that enable a user to access the functionality provided by controlled devices.

UPnP defines protocols for discovery and control of devices by CPs. UPnP does not define a streaming mechanism for use by Audio/Video devices. Some of the discovery and control protocols are part of the UPnP specification while others are separately standardized by the IETF (Internet Engineering Taks Force). Interaction between CPs and devices is based on the Internet protocol (IP). However, UPnP allows non-IP devices to be proxied by a software component running on IP-compliant devices. Such a component, called

Controlled Device (CD) proxy, is responsible for translation and forwarding of UPnP interactions to the proxied device.

A UPnP device has a hierarchy of sub-devices with at the lowest level services. Both devices and services have standardized types. A device type determines the sub-devices or services that it is allowed to contain. A service type defines the actions and state variables that a service is allowed to contain. State variables model the state of the device, actions can be invoked by a CP in order to change that state. The description of the state variables and the action is called the SCP (Service Control Protocol). A UPnP device provides a description of itself in the form of an XML document. This document contains, among other things, the service types that it supports. Optionally, a device may have a presentation server for direct UI control by a CP.

UPnP relies on AutoIP, which provides a means for an IP device to get a unique address in the absence of a DHCP server. UPnP defines a discovery protocol, based on UDP multicast, called SSDP (Simple Service Discovery Protocol). SSDP is based on devices periodically multicasting announcements of the services that they provide. An announcement contains a URL to which service actions are to be sent: the control server. In addition to that, CPs may query the UPnP network for particular device or services types or instances.

UPnP relies on GENA (Generic Event Notification Architecture) to define a state variable subscription and change notification mechanism based on TCP.

After a CP has detected a service it wants to use (via SSDP), it controls the service by sending SCP actions to the control server URL or querying for state variables. Actions are sent using HTTP POST messages. The body of these messages are defined by the SOAP (Simple Object Access Protocol) standard. SOAP defines a remote procedure call mechanism based on XML.

The translation modules, or software representations, of HAVi devices in UPnP are called Controlled Device (CD) *proxies*, while the software representations of UPnP devices in HAVi are called *Device Control Modules* (DCMs).

HAVi to UPnP bridging

Fig 2 is a block diagram of a home network system 200 illustrating the bridging from HAVi to UPnP, and shows in bold arrows the sequence of steps to bridge a HAVi device, here a digital camera, to UPnP.

System 200 has a UPnP cluster formed by devices 202, 204, and 206. Device 202 comprises a light, device 204 comprises an MP3 player, and device 206 comprises a

printer. System 200 has a HAVi network cluster formed by a TV 208, and a digital video recorder 210. The clusters are connected through bridge 118.

In a step 212 a HAVi Camera 214 is physically plugged into the HAVi's 1394 network, thus making Camera device 214 an active HAVi node.

In a step 216 this addition is discovered by the HAVi Event Manager, residing on platform components group 218. The HAVi platform listens and reacts to the HAVi NewSoftwareElement event, or listens to a HAVi NetworkReset event to discover Camera 214 as new device.

In a step 220 the registration attributes of the DCM of Camera 214 and its FCM components are retrieved from the HAVi Registry on platform 218, and are encoded in some format understood by a bridge server 222, for example XML, and are sent to bridge server 222 using HTTP POST. Bridge 118 may use a HAVi Webproxy FCM to implement this.

In a step 224 a look-up component maps a HAVi device description, in the form of DCM/FCM registration attributes, to a UPnP CD proxy 226 for that device. Since UPnP CD proxies and HAVi DCMs are composite objects, the location process might be implemented on the sub device (component) level, as described earlier. In HAVi a device (software representation is DCM) consists of a number of *functional components* (software representation is FCM). To find out which FCMs are part of a DCM bridge 118 can use the DCM::GetFcmSeidList and FCM::GetFcmType methods, or look at registry entries that have the same values for the GUID and n1 field of the TargetId attribute. In UPnP, a device is a hierarchical structure of sub devices with at the lowest level *services.* FCMs serve the same purpose as services. The mapping of HAVi device to UPnP CD proxy 226 might be from complete DCM to complete CD proxy, or partially from FCMs to proxy services. The mapping of FCM to service might be 1-to-1, 1-to-many or many-to-many.

In a step 228 downloaded CD proxy 226 is run on the execution environment of bridge 118. This involves installing an http server for the unique URL of CD proxy 226.

In a step 230 CD proxy 226 sends out periodic announcement messages, and responds to discover messages. This enables the other UPnP applications and devices to discover and use the HAVi Camera 214 through the CD proxy 226.

UPnP to HAVi bridging

Fig.3 illustrates the steps to bridge a UPnP device, here printer 206 to HAVi cluster 208, 210, 214 in system 200.

In a step 302, UPnP Printer 206 is physically plugged into the UPnP network, and 'powering-up' the UPnP device.

A next step 304 involves listening and reacting on the UPnP device announcement message.

In a step 306, the device description document of printer 206 is retrieved from the URL embodied in the announcement message, and the document is sent to bridge server 222 using HTTP POST.

A step 308 involves a look-up component that maps a UPnP device description, in the form of a description document (in XML), to a HAVi DCM for that device, here printer 206. Since UPnP CD proxies and HAVi DCMs are composite objects, the location process might be implemented on the sub device (component) level, as described earlier. In HAVi a device (software representation is DCM) consists of a number of *functional components* (software representation is FCM). In UPnP, a device is a hierarchical structure of sub devices with, at the lowest level, *services.* Services that are part of a UPnP device can be found in the device description document. FCMs serve the same purpose as services. The mapping of HAVi device to UPnP CD proxy might be from complete DCM to complete CD proxy, or partially from FCMs to services. The mapping of FCM to service might be 1-to-1, 1-to-many or many-to-many.

A step 310 involves running a downloaded Printer DCM 312 in the execution environment of bridge 118. This involves calling the DCM's Install method.

A step 314 involves DCM 312 and its FCMs to create HAVi software elements, and using those to register with the HAVi registry component (which is part of platform components 218 available on bridge 118).

In a step 316 the HAVi registry posts global NewSoftwareElement events for DCM 312 and all FCMs that are part of it. This enables the other HAVi applications and devices to discover and use UPnP Printer 206 through Printer DCM 312.

CLAIMS:

1. A method of providing a service to a user of a home network (100), wherein:
- the method comprises enabling a component (116) of a first cluster (110) in the network to interact with a second cluster (102) of the home network;
- the first cluster has a first software architecture;
- the second cluster has a second software architectures different from the first architecture;
- the first and second clusters are coupled through a bridge (118);
- the method comprises:
- enabling a server (124) external to the clusters to receive a reference of a component having a first software representation in the first cluster; and
- enabling to provide to the bridge a translation module (128), associated with the reference, for at least partially representing the component on the second cluster upon the module being installed on the bridge.

2. The method of claim 1, wherein the server receives the reference from the bridge.

3. The method of claim 1, wherein the bridge contacts the server via the Internet.

4. The method of claim 1, wherein the first cluster comprises a HAVi cluster.

5. The method of claim 1, wherein the first cluster comprises a UPnP cluster.

6. A data base comprising at least one translation module (128) for enabling a component (116) on a first home network cluster (110) of a first software architecture to interact with a second home network cluster (102) of a second software architecture upon the module having been downloaded via the Internet on a bridge (118) that couples the first and second clusters.

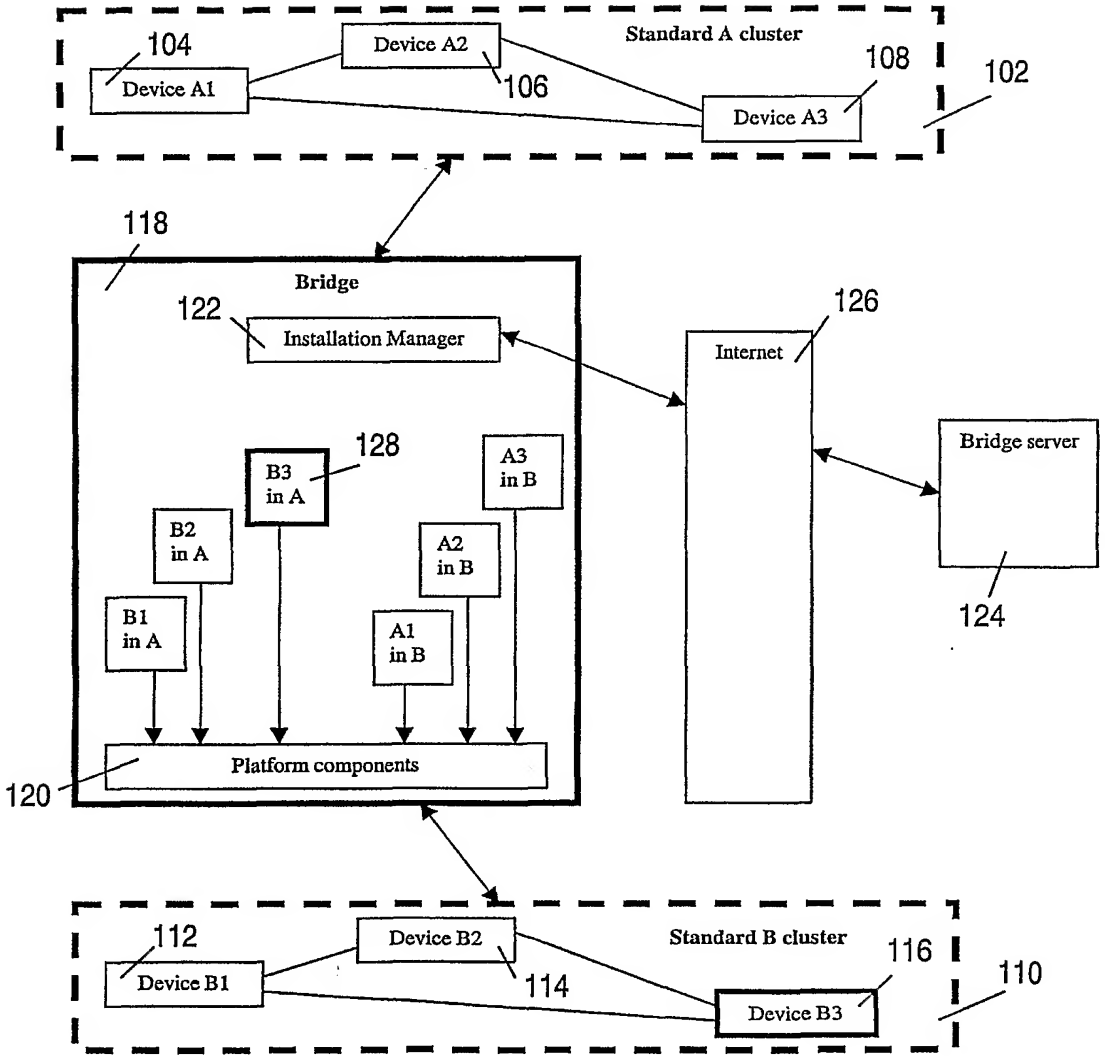
Standard A cluster
104
Device A1
Device A2
106
108
Device A3
102
118
Bridge
122
Installation Manager
126
Internet
Bridge server
124
128
B3 in A
A3 in B
B2 in A
A2 in B
B1 in A
A1 in B
Platform components
120
112
Device B1
Device B2
114
Standard B cluster
116
Device B3
110
100

FIG. 1

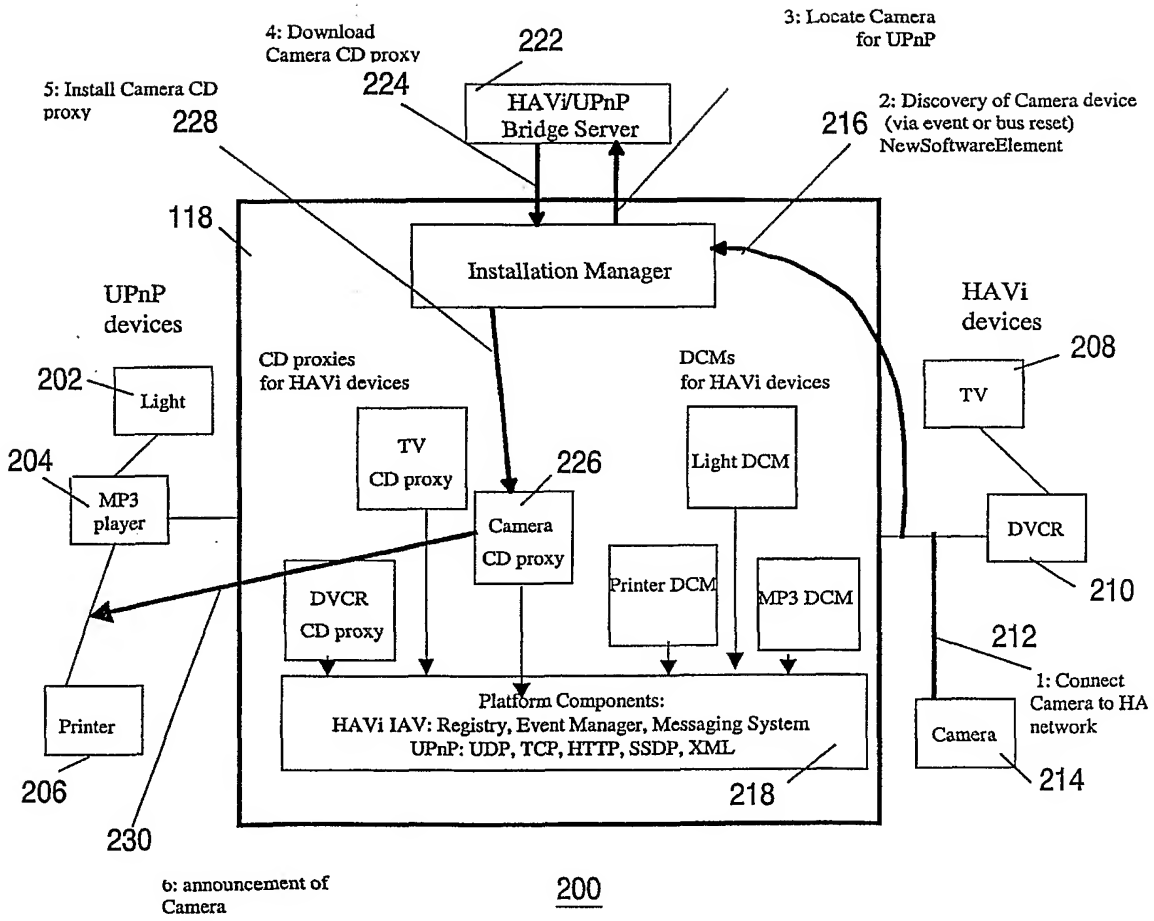
4: Download
Camera CD proxy
222
224
3: Locate Camera
for UPnP
5: Install Camera CD
proxy
228
HAVi/UPnP
Bridge Server
216
2: Discovery of Camera device
(via event or bus reset)
NewSoftwareElement
118
Installation Manager
UPnP
devices
HAVi
devices
CD proxies
for HAVi devices
DCMs
for HAVi devices
202
Light
208
TV
TV
CD proxy
226
Light DCM
204
MP3
player
Camera
CD proxy
DVCR
DVCR
CD proxy
Printer DCM
MP3 DCM
210
212
1: Connect
Camera to HA
network
Printer
Platform Components:
HAVi IAV: Registry, Event Manager, Messaging System
UPnP: UDP, TCP, HTTP, SSDP, XML
Camera
206
218
214
230
6: announcement of
Camera
200

FIG. 2

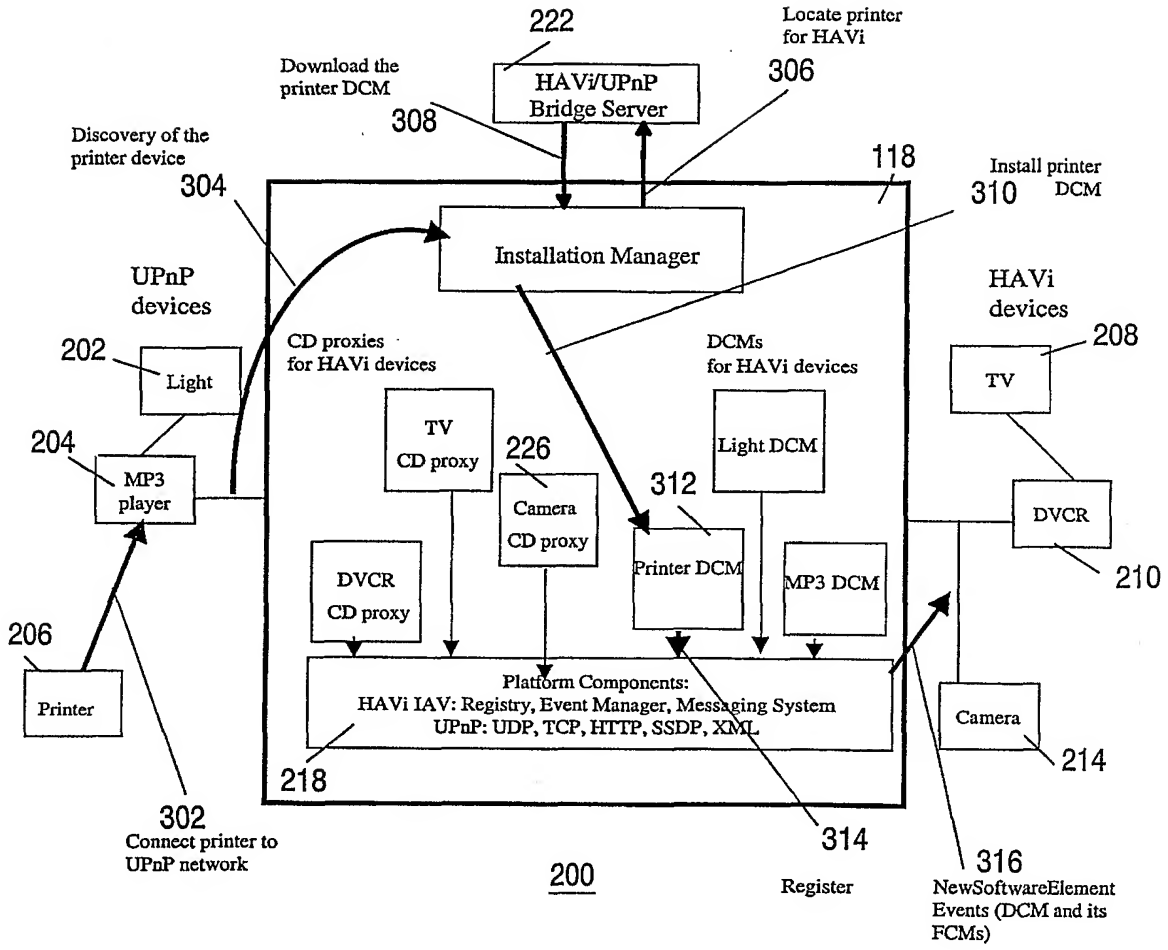
222
Locate printer for HAVi
Download the printer DCM
HAVi/UPnP Bridge Server
306
308
Discovery of the printer device
304
118
Install printer DCM
310
Installation Manager
UPnP devices
HAVi devices
CD proxies for HAVi devices
DCMs for HAVi devices
202
Light
208
TV
204
MP3 player
TV CD proxy
226
Light DCM
Camera CD proxy
312
DVCR
210
DVCR CD proxy
Printer DCM
MP3 DCM
206
Printer
Platform Components:
HAVi IAV: Registry, Event Manager, Messaging System
UPnP: UDP, TCP, HTTP, SSDP, XML
Camera
214
218
302
Connect printer to UPnP network
314
Register
316
NewSoftwareElement Events (DCM and its FCMs)
200

FIG. 3